E.N.S.E.R.B.
Romain COMBELAS
Frédéric BONNIN
3ème année. Option TIC

MÉMOIRE DE FIN D'ÉTUDES

RÉALISATION ET MISE AU POINT D'UN **SYSTÈME DE TÉLÉSURVEILLANCE VIDÉO**

Encadré par :

M. Patrice KADIONIK
M. Yannick BERTHOUMIEU

Remerciements

Nous tenons à remercier tout particulièrement nos responsables de projet, Messieurs Patrice KADIONIK et Yannick BERTHOUMIEU, pour l'aide précieuse qu'ils nous ont apportée tout au long de ces trois mois. Leur encadrement et leur expérience fut vraiment une expérience enrichissante.

Nous tenons également à adresser un grand merci :

- à tous les membres du service technique de l'ENSERB, M. LALANDE pour son aide dans la réalisation de la carte, M. MICOULEAU pour son aide sur Mentor Graphic, M. MARSAN responsable du service achat de l'ENSERB.

- à M. MARCHEGAY, responsable de la filière T.I.C.

- à M. DULAU pour son aide sur la partie vidéo.

Table des matières

TABLE DES MATIÈRES

Table des matières

1. PRÉSENTATION DU SUJET

1.1. LE SUJET

L'objectif du stage est la réalisation et la mise en oeuvre d'un système de surveillance vidéo. Une étude a déjà été menée : L'élaboration du schéma électrique ainsi qu'une plaque de test ont été validés.

Notre travail consiste en la création du circuit imprimé complet. Dans un premier temps le travail demandé est de valider le travail effectué par ceux qui ont commencé ce projet. Ensuite, il nous faudra implanter le microcontrôleur qui réalisera le traitement et le transfert des données.

Les applications du système de surveillance vidéo peuvent être :

- la lutte contre une tentative de pénétration dans une zone privée.
- la surveillance d'installations publiques ou industrielles (musées, banques, usines, etc...).

Le principe général du système de vidéosurveillance est le suivant :

- La surveillance se fera par caméras vidéo miniatures fixes disposées aux endroits stratégiques pour une meilleure détection.
- Les caméras sont reliées à un système de traitement d'images (notre carte) permettant la détection de tout changement dans la zone surveillée (présence humaine, etc...).
- Le système sera relié à un système d'alarme classique (sirène ou autre).
- Au moment de la détection de l'événement, l'image vidéo prise par la caméra sera gardée en mémoire. Elle pourra être visualisée sur un écran PC grâce à une transmission sur liaison série classique (norme RS232).
- Le système sera alimenté en secteur secouru (batteries). Les caméras seront conditionnées dans des boîtiers aptes à supporter des conditions climatiques extérieures.

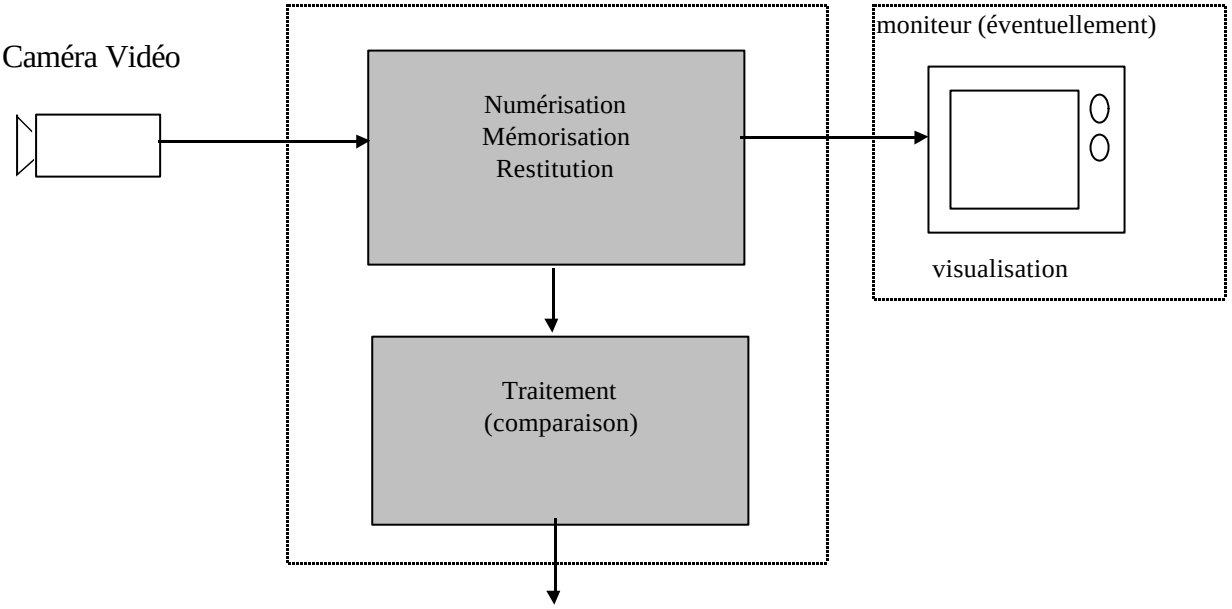
La base du système d'enregistrement consiste en la mémorisation d'images instantanées à une fréquence déterminée. Il y a comparaison d'une image I_{n+1} avec l'image I_{n0} déjà mémorisée. S'il n'y a pas de changement, l'image I_{n+1} devient image de référence I_{n0}.
Dès que la modification du paysage surveillé devient significative, il y a mémorisation définitive de l'image I_{n+1}.

1.2. ANALYSE DE L'EXISTANT

1.2.1. Introduction

La caméra fournit un signal vidéo monochrome analogique (norme CCIR) qui est numérisé et mémorisé. On effectue alors une comparaison de deux images vidéo consécutives en analysant les données échantillonnées. Lorsque la différence entre ces deux images est significative, un système d'alarme est déclenché. La Figure 1-1 page 6 représente le schéma bloc général.

1.2.2. Schéma bloc



Résultat du traitement :
Déclenchement de l'alarme,
Sauvegarde et/ou transfert de l'image utile.

Figure 1-1: Schéma bloc général.

1.2.3. Synoptique de la chaîne acquisition-restitution

Le schéma ci-dessous (Figure 1-2 page 7) représente toutes les étapes de prétraitement, d'acquisition et de mise en forme du signal vidéo.

On peut distinguer 4 grandes parties :

- la partie «séquence » pilote le CAN et le CNA et génère les cycles de lecture/écriture en fonction des ordres qu'elle reçoit,

Présentation du sujet

- la partie « numérisation » constituée par le CAN échantillonne le signal analogique en un signal numérique 8 bits,
- la partie « stockage », constituée par 2 RAM de 128 Koctets contenant chacune l'ensemble des échantillons d'une image,
- la partie « restitution » comprenant le CNA et l'électronique de remise en forme, se charge de reconstituer un signal analogique ayant les caractéristiques du signal provenant de la caméra (amplitude, offset...)

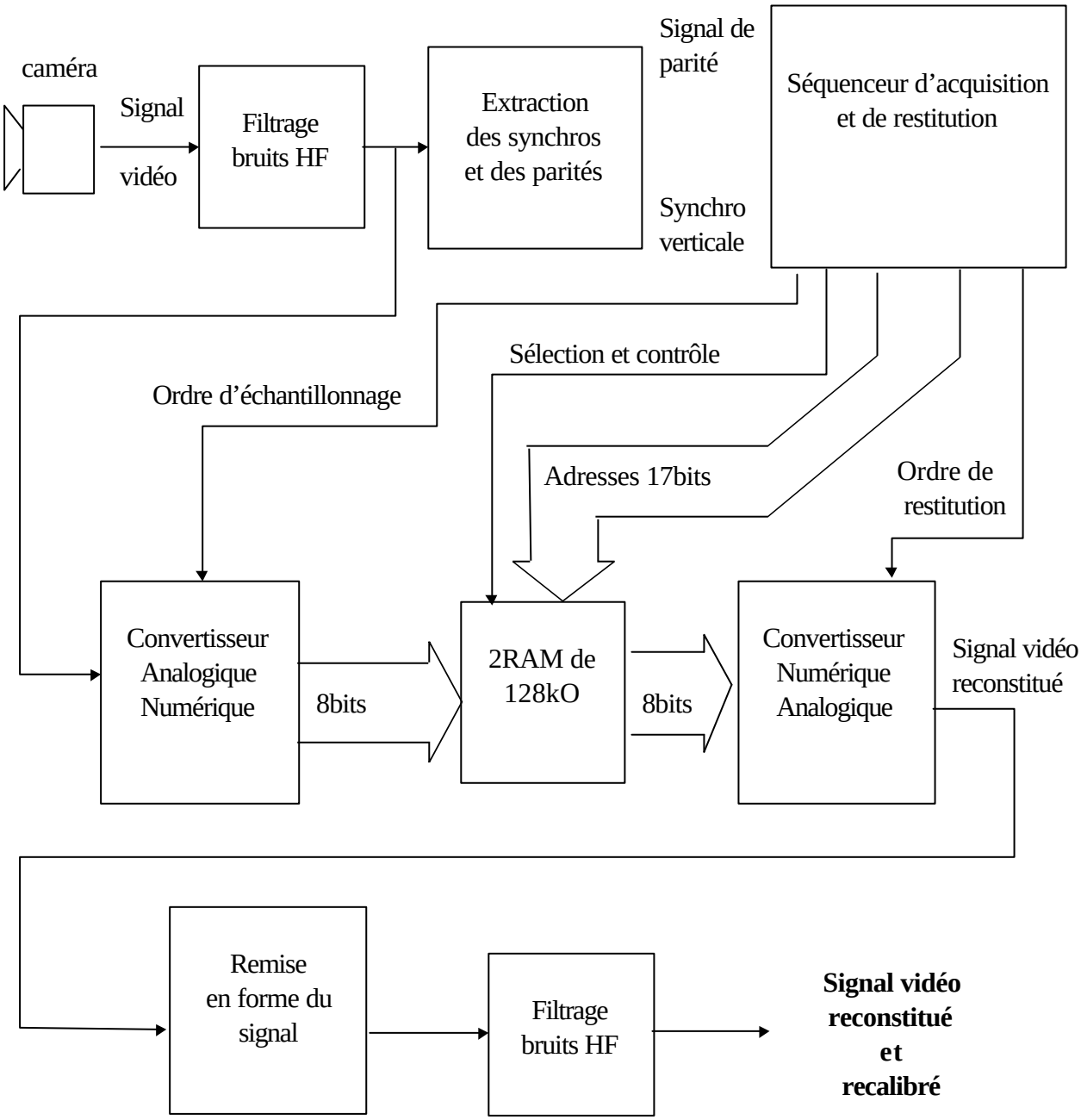


Figure 1-2 : Synoptique de la chaîne d'acquisition restitution

2. PRINCIPE DE FONCTIONNEMENT

2.1. INTRODUCTION

Cette partie a pour objectif d'expliquer les processus d'acquisition, de transfert et de restitution d'une image vidéo. La partie concernant l'analyse de l'image et donc l'intégration de l'unité de traitement sera traitée par la suite dans la partie 4. Il est conseillé de se reporter au: schéma de principe de la page 21.

On souhaite réaliser et mettre en oeuvre une carte de surveillance vidéo. Une maquette a déjà été réalisée. Notre but est de faire une carte définitive. Une première étape consistera évidemment par le routage de la carte, grâce aux outils de C.A.O. de chez Mentor Graphics, à partir d'un schéma électrique vérifié. Il nous faudra tout d'abord valider le fonctionnement sans le microcontrôleur, c'est à dire reprendre le projet où nous l'avons trouvé. Puis on intégrera le microcontrôleur et on pourra ensuite étudier et tester le traitement des images, c'est à dire la détection de présence.

2.2. ACQUISITION

2.2.1. Principe de l'acquisition

L'acquisition des images fournies par la caméra se fait successivement dans les RAM 0 et 1.
Le séquenceur est averti de la RAM dans laquelle l'image doit être stockée grâce au signal **choix_ram** (choix_ram = 0 =>stockage dans la RAM 0 ; choix_ram = 1 => stockage dans la RAM 1).
Le déclenchement de l'opération d'acquisition se fait avec la validation du signal **start_acq**. Le séquenceur attend alors le début d'une trame. Le LM 1881, directement connecté au signal vidéo fourni par la caméra est chargé de scruter le début d'une trame. Sa sortie **V_sync** passe à 0 lorsque celle-ci est détectée. Le séquenceur commence alors à numériser le signal vidéo en activant l'horloge d'un CAN (**clk_can**). Le séquenceur pilote également le compteur (grâce au signal **increm**) qui fournit à la RAM sélectionnée en écriture (grâce au signaux **cs0** ou **cs1** et **we_ram**) l'adresse dans laquelle chaque échantillon numérique à la sortie du CAN doit être stocké (le compteur ayant été remis à 0 au début de la phase d'acquisition grâce au signal **reset**).
Pour éviter la génération de signaux de synchronisation lors de la restitution de l'image à partir du signal numérisé, on a choisi d'échantillonner le signal vidéo analogique dans son intégralité, même si, les synchros lignes et trames sont inutiles pour la comparaison des images. Avec ce choix, la taille mémoire nécessaire restera néanmoins raisonnable. En effet, en considérant que la fréquence maximale d'un signal vidéo composite est de 0.5 Mhz, on obtient avec le théorème de Shannon une fréquence d'échantillonnage minimale de 1 Mhz; Par sécurité, on choisit d'échantillonner à 2 Mhz, c'est-à-dire tous les 0.5 µs. La durée d'une image vidéo étant de 40 ms, on aura besoin ainsi de 80 Koctets pour stocker une image.

Principe de fonctionnement

Le compteur incrémente donc les adresses jusqu'à atteindre 80000. Une fois arrivé, celui-ci prévient le séquenceur en validant le signal **fin_compt**, et le séquenceur valide à son tour le signal **fin_num** qui avertit que l'opération d'acquisition est terminée.

2.2.2. Programmation du séquenceur pour l'acquisition

Ci-dessous le programme AHDL du séquenceur (un Altera MAX7000 , voire la partie 3.1.1 page 22).
L'état s1 est atteint si et seulement si on a la condition : (**start_acq**) & (!**visu**) & (!**transf_ext**).
Le signal **copy** n'est pas utilisé, et les bus **count_adr[16..0]** et **adresse_out[16..0]** peuvent être assimilés à **Ad[0..16]** (leur différentiation sera justifiée par la suite, dans la partie 4.3 page 41).

Principe de fonctionnement

```
WHEN s1 =>          % Etape d'initialisation d'acquisition %

dcs0 = GND;
dcs1 = GND;
dcs2 = GND;
count_adr[ ] = 0;
/oe_ram = VCC;
/oe2 = VCC;
/oe_can = GND;
clk_acq = VCC;
/we_ram = VCC;
/we_ram2 = VCC;
fin_num = GND;
fin_transf=GND;
clk_cna = VCC;
reset = VCC;          % RAZ compteur %
increm = GND;
copy = GND;

IF (vsync) THEN
    ss = s2;
END IF;

WHEN s2 =>          % Attente du début d'une trame (passage de vsync à 0) %

dcs0 = GND;
dcs1 = GND;
dcs2 = GND;
count_adr[ ] = 0;
/oe_ram = VCC;
/oe2 = VCC;
/oe_can = GND;
clk_acq = VCC;
/we_ram = VCC;
/we_ram2 = VCC;
fin_num = GND;
fin_transf=GND;
clk_cna = VCC;
cpteur[ ] = 0;        %Compteur du nombre de coups d'horloge du can initialisé à 0 %
reset = GND;
increm = GND;
copy = GND;

IF (!vsync) THEN
    ss = s3;
END IF;

WHEN s3 =>          %Début de numérisation.Pendant les 3 1ers échantillonnages, pasd'écriture.%

dcs0 = GND;
dcs1 = GND;
dcs2 = GND;
/oe_ram = VCC;
```

Principe de fonctionnement

```
/oe2 = VCC;
/oe_can = GND;
/we_ram = VCC;
/we_ram2 = VCC;
clk_acq = GND;      % Conversion sur front descendant;Données valides au 3e front montant suivant %
count_adr[ ] = count_adr[ ];
fin_num = GND;
fin_transf=GND;
clk_cna = VCC;
cpteur[ ] = cpteur[ ];
reset = GND;
increm = GND;
arret = arret;
copy = GND;
ss = s4;

WHEN s4 =>          % 1ere étape cycle d'écriture.%

IF cpteur[ ] >= 2 THEN
        IF choix_ram THEN
            dcs0 = GND;
            dcs1 = VCC;
        ELSE
            dcs0 = VCC;
            dcs1 = GND;
        END IF;
ELSE
        dcs0 = GND;
        dcs1 = GND;
END IF;

    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    clk_acq = GND;
    /we_ram = VCC;
    /we_ram2 = VCC;
    count_adr[ ] = count_adr[ ];
    fin_num = GND;
    fin_transf=GND;
    clk_cna = VCC;
    cpteur[ ] = cpteur[ ];
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
    ss = s5;

WHEN s5 =>          % 2ème étape du cycle d'écriture. A partir d'ici, données stables %

dcs0 = GND;
dcs1 = GND;
dcs2 = GND;
```

Principe de fonctionnement

```
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
/we_ram2 = VCC;
    IF cpteur[ ] >= 2 THEN
        /we_ram = GND;
        ELSE
        /we_ram = VCC;
    END IF;

        count_adr[ ] = count_adr[ ];
        fin_num = GND;
        fin_transf=GND;
        clk_cna = VCC;
        cpteur[ ] = cpteur[ ];
        reset = GND;
        increm = GND;
        arret = arret;
        copy = GND;
        ss = s6;

    WHEN s6 =>          % Incrémentation adresse %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    clk_acq = VCC;
    /we_ram = VCC;
    /we_ram2 = VCC;
    fin_num = GND;
    fin_transf=GND;
    clk_cna = VCC;
    copy = GND;

    IF (!fin_compt) THEN

        IF cpteur[ ] >= 2 THEN
            increm = VCC;
            reset = GND;
            count_adr[ ] = val_compt[16..0];
            cpteur[ ] = 3;
        ELSIF cpteur[ ] ==1 THEN    % à cause du décalage des compteurs d'adresses%
            increm = VCC;
            reset = GND;
            count_adr[ ] = count_adr[ ];
            cpteur[ ] = 2;
        ELSE
            count_adr[ ] = count_adr[ ];
            reset = GND;
            increm = GND;
            cpteur[ ] = cpteur[ ] + H"00001";
```

Principe de fonctionnement

```
        END IF;
        arret = GND;
        ss = s3;
    ELSE
        reset = GND;
        increm = GND;
        IF (arret == GND) THEN        %il faut faire un cycle de plus%
            count_adr[ ] = val_compt[16..0];
            arret = VCC;
            cpteur[ ] = cpteur[ ];
            ss = s3;
            ELSE
                cpteur[ ] = cpteur[ ];
                count_adr[ ] = count_adr[ ];
                ss = s7;
        END IF;

    END IF;

    WHEN s7 =>                        % Arrêt numérisation %

    dcs0 = dcs0;
    dcs1 = dcs1;
    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    count_adr[ ] = 0;
    /we_ram = VCC;
    /we_ram2 = VCC;
    clk_acq = VCC;
    fin_num = VCC;
    fin_transf=GND;
    clk_cna = VCC;
    reset = VCC;
    increm = GND;
    copy = GND;
    ss = s0;
```

Principe de fonctionnement

2.3. TRANSFERT

2.3.1. Principe du transfert

L'acquisition des images vidéo fournies par la camera se fait successivement dans les RAM 0 et 1. Lorsque le microcontrôleur détecte une différence significative entre le contenu des RAM 0 et 1, l'image la plus récente est transférée dans une mémoire de sauvegarde : la RAM 2 , d'ou elle pourra être envoyée vers un moniteur.
L'ordre de transfert est fourni par le signal **transf_ext** du microcontrôleur, qui passe par le compteur en mode transparent avant de parvenir au séquenceur.
Lorsque le signal **transf_ext** est validé, le séquenceur déclenche l'incrémentation du compteur d'adresses grâce au signal **increm** qui est activé régulièrement jusqu'à ce que le compteur atteigne 80000.
A chaque adresse fournie par le compteur sur le bus d'adresses, le séquenceur sélectionne en lecture la RAM contenant la dernière image (validation de **cs0** ou **cs1** , activation de **oe_ram**), et autorise l'écriture dans la RAM 2 (validation de **cs2** et **we_ram2**). L'indication de la RAM (0 ou 1) à transférer dans la mémoire de sauvegarde est donnée par le microcontrôleur, connecté directement au séquenceur par le signal **choix_ram.**
Une fois le compteur arrivé à 80000, celui-ci envoie au séquenceur le signal **fin_compt** , et ce dernier avertit alors le microcontrôleur de la fin de l'opération de transfert en validant le signal **fin_transf**. Le séquenceur revient alors dans son état initial S0, et le microcontrôleur redevient maître des opérations.

2.3.2. programmation du séquenceur .

Ci-dessous la programmation AHDL du séquenceur traitant du transfert de la RAM 0 ou 1 vers la RAM 2. Le signal copy n'est pas utilisé, et on peut comme précédemment assimiler les bus count_adr[16..0] et adresse_out[16..0] au bus Ad[16..0].
L'état s14 est atteint si et seulement si : (!start_acq) & (!visu) & (transf_ext)

Principe de fonctionnement

```
WHEN s14 =>          % Initialisation du transfert %

    clk_acq=VCC;
    clk_cna=VCC;
    /we_ram=VCC;
    /we_ram2=VCC;
    dcs0.d=GND;
    dcs1.d=GND;
    dcs2.d=GND;
    /oe_can=VCC;
    /oe_ram=VCC;
    /oe2=VCC;
    reset=VCC;          % Remise à zero du compteur d'adresse %
    increm=GND;
    copy=GND;
    count_adr[ ]=0;
    fin_num=GND;
    fin_transf=GND;
    ss=s15;


WHEN s15 =>          % Début du cycle de transfert %
                     % On prépare la lecture pour le coup d'horloge suivant. %

    clk_acq=VCC;
    clk_cna=VCC;
    /we_ram=VCC;
    /we_ram2=VCC;
    dcs2=GND;
    /oe_can=VCC;
    /oe_ram=VCC;
    /oe2=VCC;
    reset=GND;
    increm=GND;
    copy=GND;
    count_adr[ ]=count_adr[ ];
    fin_num=GND;
    fin_transf=GND;

    IF (choix_ram) THEN
        dcs0=GND;
        dcs1=VCC;       % donnée RAM 1 en sortie %
    ELSE
        dcs0=VCC;       % donnée RAM 0 en sortie %
        dcs1=GND;
    END IF;

    ss=s16;


    WHEN s16 =>         % lecture dans une des RAM 0 ou 1 %

    dcs0=dcs0;
```

Principe de fonctionnement

```
dcs1=dcs1;
/oe2=VCC;
/we_ram2=VCC;
dcs2=VCC;              % On prépare l'écriture dans la RAM2 %
clk_acq=VCC;
clk_cna=VCC;
/we_ram=VCC;
/oe_can=VCC;
/oe_ram=GND;           % On permet de lire sur RAM 0 ou 1 %
reset=GND;
increm=GND;
copy=GND;
count_adr[ ]=count_adr[ ];
fin_num=GND;
fin_transf=GND;
ss=s17;


WHEN s17 =>        % écriture RAM2  %

cpteur[ ]=0;
clk_acq=VCC;
clk_cna=VCC;
/oe_can=VCC;
fin_num=GND;
/we_ram=VCC;
/we_ram2=GND;
/oe_ram=GND;
/oe2=VCC;
dcs0=GND;              % Pour le coup d' H suivant %
dcs1=GND;
dcs2=GND;
copy=GND;
reset=GND;
IF (!fin_compt) THEN
    increm=VCC;
    count_adr[ ]=val_compt[16..0];
    fin_transf=GND;
    ss=s15;
ELSE
    increm=GND;
    count_adr[ ]=val_compt[16..0];
    fin_transf=VCC;
ss=s0;
END IF;
```

Principe de fonctionnement

2.4. RESTITUTION

2.4.1. Principe de la restitution

L'opération de visualisation de l'image contenue dans la RAM 2 est déclenchée par le signal **visu** provenant du microcontrôleur , et connecté directement au séquenceur.

Quand le signal **visu** est activé, le séquenceur met alors en route le compteur d'adresses en l'incrémentant régulièrement de 1 à 80000 (signal **increm**). A chaque nouvelle adresse contenue sur le bus d'adresse, le séquenceur sélectionne la RAM 2 en lecture (validation de **cs2** et **oe2**), et déclenche le front montant d'un convertisseur numérique analogique.
A la fin du comptage des adresses, le compteur envoie le signal **fin-compt** au séquenceur ; si le signal visu est toujours actif, alors le séquenceur redéclenche une nouvelle opération de lecture-conversion de la RAM 2, sinon le séquenceur revient dans son état initial S0, et le microcontrôleur redevient maître des opérations.

Le signal analogique reconstitué à la sortie du TDA va devoir être remis en forme pour pouvoir être visualisé sur le moniteur. Le signal affaibli doit tout d'abord être amplifié : c'est le rôle de l'amplificateur différentiel à gain variable qui se trouve à la sortie du convertisseur (la perte de la composante continue par les capacités situées à l'entrée de l'amplificateur est sans importance puisque le moniteur est capable de la restituer automatiquement).Un filtre passe-bas RC permet ensuite d'éliminer les parasites HF (fréquence > 500 kHz), et l'adaptation d'impédance avec le moniteur est réalisée avec un montage suiveur suivi d'une résistance de 75 ohms.

2.4.2. Programmation du séquenceur

Le séquenceur se rend à l'état s8 si et seulement si : (!start_acq) & (visu) & (!transf_ext)

Principe de fonctionnement

```
WHEN s8 =>          % visualisation de l'image numérisée %

dcs0 = GND;
dcs1 = GND;
dcs2 = VCC;
/oe_ram =VCC ;
/oe2 = GND;
/oe_can = VCC;
count_adr[ ] = 0;
/we_ram = VCC;
/we_ram2 = VCC;
clk_acq = VCC;
fin_num = GND;
fin_transf=GND;
clk_cna = VCC;
reset = VCC;
increm = GND;
copy = GND;
ss = s9;

WHEN s9 =>          % étape1 de visualisation %

dcs0 = GND;
dcs1 = GND;
dcs2 = VCC;
/oe_ram =VCC;
/oe2 = GND;
/oe_can = VCC;
count_adr[ ] = count_adr[ ];
/we_ram = VCC;
/we_ram2 = VCC;
clk_acq = VCC;
fin_num = GND;
fin_transf=GND;
clk_cna = VCC;
reset = GND;
increm = GND;
arret = arret;
copy = GND;
ss = s10;

WHEN s10 =>         % étape2 de visualisation %

dcs0 = GND;
dcs1 = GND;
dcs2 = VCC;
/oe_ram = VCC;
/oe2 = GND;
/oe_can = VCC;
    count_adr[ ] = count_adr[ ];
    /we_ram = VCC;
    /we_ram2 = VCC;
    clk_acq = VCC;
```

Principe de fonctionnement

```
    fin_num = GND;
    fin_transf=GND;
    clk_cna = GND;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
    ss = s11;

WHEN s11 =>        % étape3 de visualisation => conversion sur front descendant de clk_cna%

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = VCC;
    /oe_ram = VCC;
    /oe2 = GND;
    /oe_can = VCC;
    count_adr[ ] = count_adr[ ];
    /we_ram = VCC;
    /we_ram2 = VCC;
    clk_acq = VCC;
    fin_num = GND;
    fin_transf=GND;
    clk_cna = GND;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
    ss = s12;

WHEN s12 =>        % étape4 de visualisation %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = VCC;
    /oe_ram = VCC;
    /oe2 = GND;
    /oe_can = VCC;
    /we_ram = VCC;
    /we_ram2 = VCC;
    clk_acq = VCC;
    fin_num = GND;
    fin_transf=GND;
    clk_cna = VCC;
    copy = GND;

    IF (visu) & (!fin_compt) THEN % visu sans être au bout d'une image %

        ss = s9;
        count_adr[ ] = val_compt[16..0];

        reset = GND;
        increm = VCC;
        arret = GND; %bit de test d'arrêt initialisé%
```

Principe de fonctionnement

```
    ELSIF (visu) & (fin_compt) THEN % fin d'une image, continuation visu %
        IF (arret == GND) THEN          %un cycle supplémentaire%
            ss = s9;
            count_adr[ ] = val_compt[16..0];
            reset = GND;
            increm = VCC;
            arret = VCC;                    %bit d'arrêt on%
        ELSE
            ss = s13;
            count_adr[ ] = 0;               % réinit de tous les compteurs%
            reset = VCC;
            increm = GND;
        END IF;
    ELSE
        count_adr[ ] = count_adr[ ];
        reset = GND;
        increm = GND;
        ss = s0;

    END IF;

WHEN s13 => % rebouclage pour visualisation %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = VCC;
    /oe_ram = VCC;
    /oe2 = GND;
    /oe_can = VCC;
    count_adr[ ] = 0; % RAZ du compteur adresse %
    /we_ram = VCC;
    /we_ram2 = VCC;
    clk_acq = VCC;
    fin_num = GND;
    fin_transf=GND;
    clk_cna = VCC;
    reset = GND; % lors du rebouclage pb de decalage des adresses %
    increm = VCC;
    arret = GND;
    copy = GND;

    IF (visu) THEN
        ss = s10; % pour respecter le timing %
    ELSE
        ss = s0;
    END IF;
```

Principe de fonctionnement

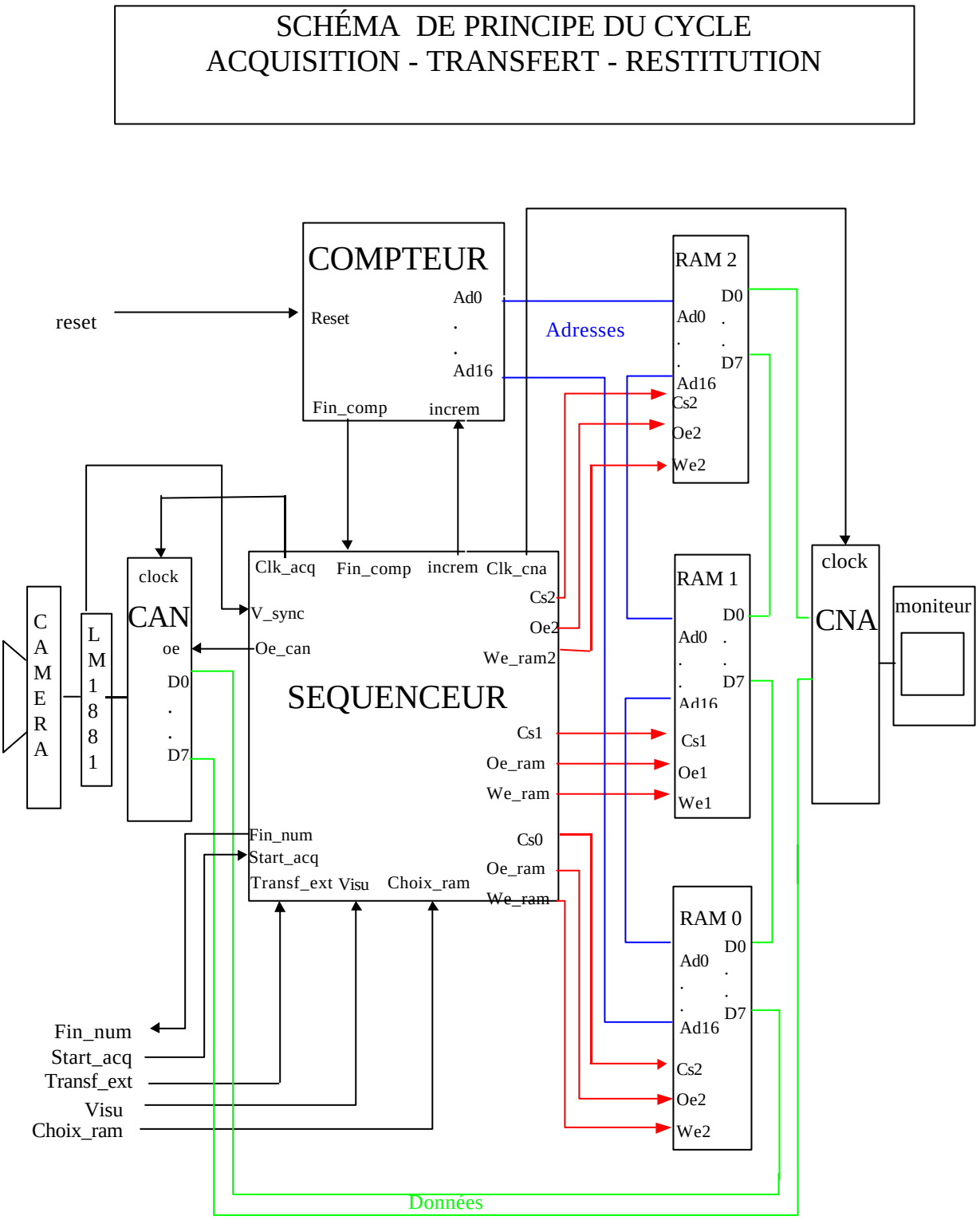


Figure 2-1 : schéma de principe

3. RÉALISATION ET SIMULATION

3.1. COMPOSANTS

3.1.1. La famille ALTERA MAX7000

3.1.1.1.Présentation :

Les circuits de la famille MAX7000 d'ALTERA sont des circuits logiques reprogrammables (EPLD). La configuration est stockée de manière permanente grâce à la technologie EEPROM qui permet environ 100 cycles d'effacement-programmation.
Ces circuits sont donc très adaptés aux applications prototypes, ce qui est précisément notre cas.

3.1.1.2.Description fonctionnelle :

L'architecture des circuits de cette famille est organisée en cellules qui comprennent une partie logique combinatoire, des registres et des blocs d'entrée-sortie. Un circuit compte des cellules internes ainsi qu'une cellule pour chaque patte d'entrée-sortie.
L'utilisateur développe son circuit en utilisant le logiciel fourni par le fabricant, *MAX+plusII*. A l'aide d'un schéma ou d'une description fonctionnelle de l'application (en AHDL), un compilateur spécialisé génère un code pour la programmation du circuit. L'utilisateur ne tient compte que du nombre de portes disponibles et de la fréquence max. de fonctionnement du circuit.
Par exemple, le circuit EPM7096LC84 comporte 160 cellules et 84 broches.

3.1.2. Les convertisseurs

3.1.2.1.Le CAN

Les caractéristiques du signal vidéo provenant de la caméra sont :

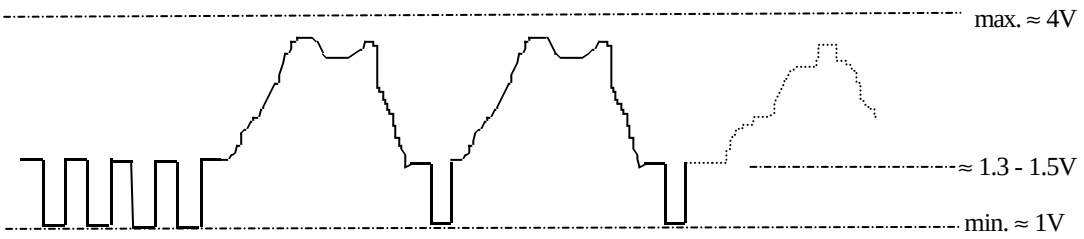


Figure 3-1 : Caractéristiques du signal vidéo.

On choisit donc un convertisseur flash 8 bits dédié à la vidéo. La plage de conversion, [1..4]V pour nous, est fixée par les entrées V_{RB} et V_{RT} respectivement. L'entrée _OE est contrôlée par le séquenceur.
Il convertit sur front descendant de l'horloge et les données sont disponibles en sortie après le 3ème front montant suivant (ceci aura son importance pour le séquenceur).

3.1.2.2.Le CNA

Le choix s'est porté sur le TDA8702 de Phillips spécialement conçu pour la vidéo (impédance de sortie de 75Ω).

Il possède 2 modes de fonctionnement :

- Durant le « transparent mode » (CLK niveau bas), tout changement de données en entrée sera convertit en sortie analogique.
- Pendant le « latched mode », (CLK niveau haut), le signal analogique reste stable indépendamment de tout changement en entrée.

Les tensions analogiques en sortie (V_{out} et $\_V_{out}$) sont référencées par rapport à V_{CCA} (tension de référence max. pour le signal analogique). Avec une impédance de 75Ω vue par la sortie du convertisseur, le code 0 en entrée correspond à une sortie :

$$V_{out} = V_{CCA} = 5V$$

et le code 255 à une sortie :

$$V_{out} = V_{CCA} - 0.8V = 4.2V.$$

La tension $\_V_{out}$ étant la tension complémentaire de V_{out}.

3.2. SIMULATION

Les simualtions effectuées ont été réalisées sans le microcontrôleur, de manière à valider la carte au même niveau de ce qui a déjà été réalisé, c'est à dire l'acquisition et la visualisation du signal vidéo. Le seul changement est qu'il y a en plus un transfert des données, de la RAM0 ou de la RAM1 vers la RAM2, car lorsque le microcontrôleur sera rajouté, la visualisation ne pourra se faire qu'à partir de la RAM2.

Les six courbes qui suivent ont été réalisées grâce à l'analyseur logique Hewlett Packard 16500A.

Les sept courbes d'aprés ont été réalisées sur un oscillateur METRIX OX 8627 100mhz.

La Figure 3-8, page 30, est une simulation effectuée par le binôme qui s'est occupé du projet juste avant nous. Elle a été réalisée sur une plaquette d'essai. On peut constaté que nous avons apporté une amélioration significative au système car le signal est nettement moins bruité.

Figure 3-2 : Début d'acquisition.

Nous avons activé manuellement le signal start_acq, puis le signal vsync (en remplacement du signal de synchronisation vertical sur le signal vidéo). L'analyseur logique est déclenché sur le front descendant de vsync.
Le séquenceur atteint bien l'état s1 où l'acquisition commence. Le fonctionnement est tout à fait normal : clk_acq (l'horloge du CAN) se met en marche (fréquence = 2 MHz)dès que vsync passe à 0, le compteur d'adresse commence à compter dés que increm passe à 1, et l'écriture dans la RAM0 ou RAM1 se fait à partir du troisième coup de l'horloge du CAN.

Figure 3-3 : Fin d'acquisition.

L'analyseur logique est déclenché sur le front montant du signal fin_compt.
Le signal fin_compt passe à l'état 1 lorsque le compteur d'adresse atteint sa valeur de butée. L'horloge CAN, clk_acq, devient inactive et l'écriture dans la ram s'arrête. A ce moment, le séquenceur envoie un signal de fin : fin_num, et revient dans son état d'attente.

Réalisation et simulation

Figure 3-4 : Début de transfert.

Le signal transf_ext est activé manuellement et l'analyseur logique est déclenché sur le front montant de transfert_ext.
Le compteur d'adresse se déclenche quand increm passe à 1. A chaque changement d'adresses, la RAM0 ou 1 est sélectionnée, et ses données sont présentes en sortie, et la RAM2 est selectionnée et en mode écriture.
Le signal copy est l'indicateur qui permet de voir si le séquenceur sort de son état d'attente car il est à 1 uniquement dans l'état s0.

Figure 3-5 : Fin de transfert.

L'analyseur logique est déclenché sur le front montant du signal fin_compt.
Le transfert se fait correctement car le compteur d'adresse arrive à son adresse de butée et les signaux cs2, /we2, cs0 et /oe0 fonctionnent bien. Le séquenceur repasse bien dans son état d'attente dès que fin_compt passe à 1. Il n'y reste pas et repart dans un cycle de transfert tout simplement parce que le signal transf_ext est toujours présent (il n'a pas été possible de faire autrement car c'est un signal déclenché manuellement et donc moins rapide qu'un cycle de transfert).

Figure 3-6 : Début de visualisation.

Le processus de visualisation démarre lorsque le signal visu est activé manuellement. Le séquenceur quitte alors l'état 0 (copy passe à l'état bas). Le compteur d'adresse se met en marche lorsque le signal increm est au niveau 1. La RAM2 est selectionnée, et les données sont présentes sur son bus de sortie. L'horloge du CNA démarre, et à chaque changement d'adresse, les données sont converties.

Figure 3-7 : Fin de visualisation.

L'analyseur logique est déclenché sur le front descendant du signal visu.
Lorsque le signal visu passe à 0, le compteur d'adresse s'arrête, ainsi que l'horloge du CNA, la RAM2 n'est plus sélectionnée, et le séquenceur repasse dans son état d'attente (visu à 1).

Figure 3-8 : Courbe réalisée par nos prédecesseurs.

On voit que le signal restitué, pour un signal d'entrée de 1kHz, est relativement bruité. La courbe suivante a été réalisée par nos soins et montre l'amélioration apportée au système.

Figure 3-9 : Signal de sortie pour un signal d'entrée sinosoï dal de 1kHz.

En haut : sur la voie 2, le signal d'entrée de fréquence 1kHz, compris entre 1 et 3V (sonde ×10).
En bas : sur la voie 1, le signal de sortie restitué, compris entre -0.8 et 0.8V (la composante continue est enlevée par une liaison capacitive).

Réalisation et simulation

Figure 3-10 : Signal de sortie pour un signal d'entrée sinosoï dal de 10kHz.

En haut : sur la voie 2, le signal d'entrée de fréquence 10kHz, compris entre 1 et 3V (sonde ×10).
En bas : sur la voie 1, le signal de sortie restitué, compris entre -0.8 et 0.8V.

Figure 3-11: Signal de sortie pour un signal d'entrée sinosoï dal de 100kHz.

En haut : sur la voie 2, le signal d'entrée de fréquence 100kHz, compris entre 1 et 3V (sonde ×10).
En bas : sur la voie 1, le signal de sortie restitué, compris entre -0.7 et 0.7V.

Réalisation et simulation

Figure 3-12 : Signal de sortie pour un signal d'entrée carré de 1kHz.

En haut : sur la voie 2, le signal d'entrée de fréquence 1kHz, compris entre 1 et 3V (sonde ×10).
En bas : sur la voie 1, le signal de sortie restitué, compris entre -0.7 et 0.7V.

Figure 3-13 : Signal de sortie pour un signal d'entrée carré de 10kHz.

En haut : sur la voie 2, le signal d'entrée de fréquence 10kHz, compris entre 1 et 3V (sonde ×10).
En bas : sur la voie 1, le signal de sortie restitué, compris entre -0.7 et 0.7V.

Figure 3-14 : Signal de sortie pour un signal d'entrée carré de 100kHz.

En haut : sur la voie 2, le signal d'entrée de fréquence 100kHz, compris entre 1 et 3V (sonde ×10).
En bas : sur la voie 1, le signal de sortie restitué, compris entre -0.7 et 0.7V.

Figure 3-15 : Influence de l'endroit où est injecté le signal d'entrée. A gauche le signal est injecté sur l'entrée de la carte; à droite, le signal est injecté directement à l'entrée du CAN.

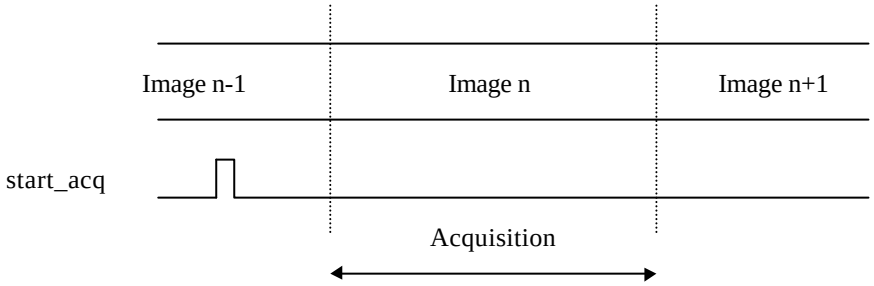
Les deux courbes représente le signal d'entrée.
Lorsque l'on injecte le signal loin de l'entré du CAN (comme c'est le cas sur notre carte), en passant à travers la partie numérique, on remarque que le signal est fortement bruité. Alors qu'en le connectant directement sur l'entrée CAN il est nettement moins bruité.

4. L'UNITÉ DE TRAITEMENT

4.1. LES SIGNAUX DE COMMANDES, LES FONCTIONNALITÉS

Les signaux de commandes sont les suivants :

❑ **start_acq :**
Impulsion qui donnera l'ordre au séquenceur de faire une acquisition. Dés qu'il passe au niveau haut (Vcc), le séquenceur bascule de l'état initial (d'attente) à un état d'attente de début de l'image qui arrive.



❑ **choix_ram :**
Signal indiquant la bank (voir chapitre 3.2) dans laquelle on souhaite mémoriser l'image.
Si choix_ram = 0 ⇒ stockage dans RAM0
sinon ⇒ stockage dans RAM1
Choix_ram est en réalité le bit XA17 provenant du microcontrôleur.

❑ **visu :**
Signal qui ordonne la restitution sur moniteur de l'image contenue dans la RAM2 tant qu'il est égal à Vcc.

❑ **transfert_ext :**
Signal envoyé du microcontrôleur au séquenceur en passant par le compteur, et indique que l'on veut effectuer le transfert de la RAM0 ou RAM1 vers la RAM2. Transfert_ext est actif lorsqu'il y a une détection de présence sur une image, ou lorsqu'on passe en mode de visualisation (qui se fait à partir de la RAM2).

❑ **fin_num :**
Lorsque l'acquisition est terminée, le séquenceur envoie un signal de fin de numérisation au microcontrôleur qui reprend alors la main.

❑ **fin_transf :**

L'unité de traitement

Lorsque le transfert de la RAM0 ou 1 vers la RAM2 est terminé, le séquenceur envoie au microcontrôleur un signal de fin de transfert.

Dans l'étude initiale on se servait de deux signaux de synchronisation d'acquisition :

❑ **o/e :**
Signal de parité de la trame actuelle (odd/even).

❑ **Vsync :**
Signal de synchronisation de trame.

Ces deux signaux étaient fournis par le LM1881 , directement connecté au signal vidéo de la camera. Or on s'est aperçu que le signal o/e était inutile : Pour la restitution , il suffit de se synchroniser sur le début d'une trame, paire ou impaire peu importe.

Les signaux de synchronisation avec le compteur d'adresses sont les suivants :

❑ **reset :**
Remise à zéro du compteur d'adresses.

❑ **reset_all :**
Remise à zéro au reset initial des deux altéras.

❑ **increm :**
Signal envoyé par le séquenceur vers le compteur , et commandant l'incrémentation de ce dernier.

❑ **fin_compt :**
Signal provenant du compteur. Impulsion délivrée quand la butée du compteur est atteinte.

❑ **copy :**
Signal généré par le séquenceur en mode transparent pour dire au compteur de recopier en sortie les adresses qu'il a en entrée. C'est le microcontrôleur qui est maître du bus d'adresses.

Le but de l'unité de traitement est d'effectuer le traitement des images ainsi que de gérer tous les signaux de contrôle. De plus, le microcontrôeur doit pouvoir adresser les trois RAM, c'est à dire 3×128 Koctets. Or un microcontrôleur classique MC68HC11 avec 16 bits d'adresses ne peut effectuer un tel adressage. C'est la raison pourquoi nous avons choisi la version K1 (MC68HC11K1), qui peut gérer cet adressage grâce à son fonctionnement en adressage paginé, développé au paragraphe suivant.

4.2. PRÉSENTATION DU MICROCONTRÔLEUR 68HC11K1

Le microcontrôleur MC68HC11 de Motorola (voir documents en annexes) est un circuit CMOS de haute technologie.

L'unité centrale 8 bits de cette famille est très voisine de celle du microprocesseur M6800.

Le MC68HC11K1 possède une EPROM de 640 octets, une RAM de 768 octets pouvant être sauvegardée par pile, ainsi que six ports multifonctions. Il possède aussi un convertisseur analogique 8 bits, des liaisons séries synchrones et asynchrones ...

Cette famille de microcontrôlleur traite les périphériques d'entrées et de sorties comme des octets mémoires ce qui signifie qu'il n'y a pas d'instructions spécifiques d'entrées/sorties pour accéder à ses périphériques.
Cette remarque est très importante car elle permet de comprendre le principe de la mémoire paginée.

La particularité du 68HC11 est de posséder un port G. Celui-ci possède des signaux « Memory Expansion » (XA13, XA14, XA15, XA16, XA17, XA18) destinés à compléter les bits d'adresses classiques.

On peut ainsi s'affranchir de la limite physique des 64 Koctets d'espace adressable. La mémoire paginée va réserver une zone d'espace physique (c'est à dire comprise dans l'espace du microcontrôleur) appelée WINDOW, ceci afin d'augmenter l'espace adressable.

Le principe est donc de diviser l'extension de mémoire en autant de pages nécessaires à la lecture ou l'écriture de toutes les adresses.
La taille de celle-ci est bien évidement égale à celle de la WINDOW.

Remarque : la taille et le nombre de pages sont imposés par le constructeur du microcontrôleur. L'utilisateur peut alors choisir une des huit possibilités qui lui sont proposées.

On veut adresser une mémoire vive de 128 Koctets. On va par exemple choisir une WINDOW de 32 Koctets (espace physique réservé) et diviser la mémoire en quatre pages de 32 Koctets chacune. Pour écrire en un lieu donné de la mémoire vive, il suffit de connaître la correspondance page/adresse mémoire vive.

4.3. L'IMPLANTATION TECHNOLOGIQUE

Pour ajouter le microcontrôleur sur la carte, il a fallu l'adapter à la partie déjà validée. C'est pourquoi la conception du schéma électrique peut paraître quelque peu étrange.

L'unité de traitement donne les ordres au séquenceur. Lorsque les ordres d'acquisition, de transfert ou de restitution sont lancés, le séquenceur prend la main, c'est à dire qu'il contrôle tous les signaux de gestion des mémoires, des convertisseurs et du bus d'adresses. L'unité de traitement doit attendre la fin des opérations du séquenceur (repérable grâce aux signaux fin_num et fin_transf) avant d'effectuer un quelconque traitement ou de lancer un nouvel ordre.

Lorsque le séquenceur est dans son état d'attente (aucun ordre reçu de la part du microcontrôleur : état s0) , c'est le MC68HC11K1 qui est maître des signaux de gestion et du bus d'adresses. Le séquenceur ne se contente alors que de recopier les signaux (CSGP1 , CSGP2 , adresse_out[ ] étant fonction des adresse_in[ ]). On appelle ce fonctionnement le **mode transparent**. Étant donné qu'il faut une synchronisation entre ces signaux , sachant que les adresses traversent les deux ALTERA, on a décidé que les signaux CSGP1 (sélection de la window1) et CSGP2 (sélection de la window2) traverseraient aussi les deux ALTERA pour avoir les mêmes temps de retard (c'est la raison de la présence des signaux C1_in, C1_out,.C2_in et C2_out.).

Le traitement des images se fait pendant le mode transparent. C'est le microcontrôleur qui gère l'accès aux RAMs. Dans l'état s0 du séquenceur, il faut donc valider, par exemple, la RAM0 si la WINDOW1 est sélectionnée et que le choix_ram se porte sur la RAM0. Le séquenceur est donc « transparent ».

Lorsque le séquenceur est dans son état d'attente les adresses en sortie (adresse_out[16..0]) suivent les variations des adresses provenant du microcontrôleur en utilisant un décodage adéquat. On a:

Si C1=1 (sélection de la Window1) alors
adresse_out[13..0] = adresse_in[13..0];
adresse_out14 = XA14;
adresse_out15 = XA15;
adresse_out16 = XA16;
choix_ram = XA17;
Si C2=1 (sélection de la Window2) alors
adresse_out[12..0] = adresse_in[12..0];
adresse_out13 = XA13;
adresse_out14 = XA14;
adresse_out15 = XA15;
adresse_out16 = XA16;Les adresses obtenues à la sortie du séquenceur permettent d'accéder aux mémoires.

Le schéma électrique complet se trouve page suivante.

L'unité de traitement

4.4. PROGRAMMATION DU MICROCONTRÔLEUR

Nous avons écris les programmes en C et en assembleur, et les avons compilés grâce au compilateur croisé pour microcontrôleur motorola MC68HC11.

A partir d'un programme écrit en C (.c), le compilateur le convertit en un fichier en assembleur (.s), puis en fichier objet (.o), puis en un fichier exécutable (.h11). Il faut ensuite lier les fichiers utiles grâce à l'éditeur de lien (.lnk) et enfin il faut les convertir en un fichier hexadécimal (.hex) pour être ensuite chargé dans la ROM. L'émulateur d'EPROM n'acceptant que le format binaire nous avons de plus dû convertir le fichier au format hexadécimal en un fichier au format binaire (.bin) grâce à un utilitaire « hex2bin ».
Un fichier de commande nommé « video » permet d'exécuter les opérations ci-dessus.

Les différents fichiers utiles sont :

- crts.s : fichier de démarrage (C runtime startup)
- init.s : fichier d'initialisation des registres du microcontrôleur
- video.c : programme principal
- vector.c : table des vecteurs d'interruptions
- video.lnk : fichier de liens

Le fichier crts.s est la routine de démarrage. C'est elle qui fait appel au programme d'initialisation et au programme principal (voir Figure 4-1, page 44). Le fichier video.lnk est le fichier de lien. C'est lui qui place les différents programmes aux endroit physique de la mémoire. Enfin le fichier vector.c contient la table des vecteurs d'interruptions. Dans notre application, nous ne nous servons pas des interruptions, donc la table des vecteurs d'interruptions ne contient rien sauf dans sa dernière adresse ($7FFE : adresse du reset). En effet à la mise sous tension, le microcontrôleur vient lire ce qui se trouve à l'adresse $7FFE. Nous avons donc placé l'adresse $A000, adresse de la ROM, dans le vecteur RESET de la table.

Quand le microcontrôleur envoie $A000 sur son bus d'adresses, la ROM, qui a un bus d'adresses de 15 bits, voit en réalité $2000 :

	a15	a14	a13	a12	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0
$A000 ⇒	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$2000 ⇒		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Il faudra donc que le programme se trouve à l'adresse $2000 dans la ROM.

L'unité de traitement

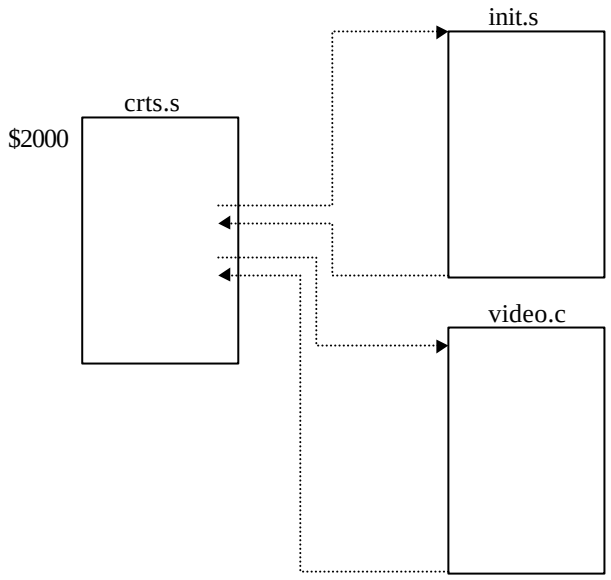


Figure 4-1 : Connections entre les différents modules

L'initialisation du microcontrôleur se fait grâce au fichier INIT.S ci-dessous : C'est elle qui met en place la configuration de la mémoire (ROM, RAM, registres, ...) ainsi que les modes de fonctionnement du microcontrôleur.

Fichier INIT.S :

```
.public  _init
         .external   _text
         .psect      _text
_init:

         .DEFINE INIT = 3DH
         .DEFINE TMSK2 = 24H
         .DEFINE CONFIG = 3FH
         .DEFINE BPROT = 35H
         .DEFINE OPTION = 39H
         .DEFINE INIT2 = 37H
         .DEFINE OPT2 = 38H
         .DEFINE CSTL = 5BH
         .DEFINE MMSIZ = 56H
         .DEFINE MMWBR = 57H
         .DEFINE MM1CR = 58H
         .DEFINE MM2CR = 59H
         .DEFINE GPCS1A = 5CH
         .DEFINE GPCS1C = 5DH
         .DEFINE GPCS2A = 5EH
         .DEFINE GPCS2C = 5FH
         .DEFINE PGAR = 2DH
```

L'unité de traitement

```
        .DEFINE HPRIO = 3CH
        .DEFINE PPAR = 2CH
;
;*******************************************************************************/
        ldaa    #0
        staa    INIT;       Les registres sont dans les 1er 128 octets
;                           et La RAM statique a partir de $80
        staa    TMSK2;      Interruptions non autorisees

        ldaa    #0CDH;      Localisation de la ROM de $A000 a $FFFF
        staa    CONFIG;     et xout disable, pas de ROM interne
        ldaa    #0FFH
        staa    BPROT;      Protection des differents blocs de l'EEPROM
        ldaa    #50H
        staa    OPTION
;*******************************************************************************/
        lds     #037H;      Haut de la pile LIFO
;*******************************************************************************/
        ldaa    #0
        staa    INIT2;      EEPROM de d80 a 0fff

        ldaa    #0
        staa    OPT2

        ldaa    #05
        staa    CSTL;       CSPROG valide pour une ROM de 32 koctets
;*******************************************************************************/
        ldaa    #26H
        staa    HPRIO;      Mode etendu

        ldaa    #0D2H
        staa    MMSIZ;      CSGP1 et 2 valides avec des pages de 16 et 8k

        ldaa    #24H
        staa    MMWBR;      adresses de base window1=$4000 windows2=$2000

        ldaa    #0
        staa    GPCS1A
        ldaa    #38H
        staa    GPCS1C

        ldaa    #0
        staa    GPCS2A

        ldaa    #37H
        staa    GPCS2C
;*******************************************************************************/
        ldaa    #1FH
        staa    PGAR;       bits XA13-17 valides

        ldx     #7000H
        jmp     1,x
;
        rts
```

L'unité de traitement

```
        .end
```

Fichier CRTS.S

```
        .LIST  +
        .external   _main, _init, __memory
        .public     _exit, __stext
;
        .psect      _bss
__sbss:
        .psect      _text
__stext:
        sei
        jsr         _init
        jsr         _main;          execute main
_exit:
        bra         _exit;          and stay here
;
        .end
```

Fichier VIDEO.LNK

```
+h                          # multi-segment output
+map=video.map
-o video.h11                # output file name
+text -b 0x2000             # program start address
+data -b 0x2300             # data start address

crts.o                      # startup routine
init.o                      # initialisation program
video.o                     # application program
c:/68hc11/lib/libi.h11      # C library (if needed)
c:/68hc11/lib/libm.h11      # machine library

+text -b 0x7fd6             # vectors start address

vector.o                    # interrupt vectors
+def __memory=__bss__       # symbol used by library
```

4.5. CARTOGRAPHIE DE LA MÉMOIRE

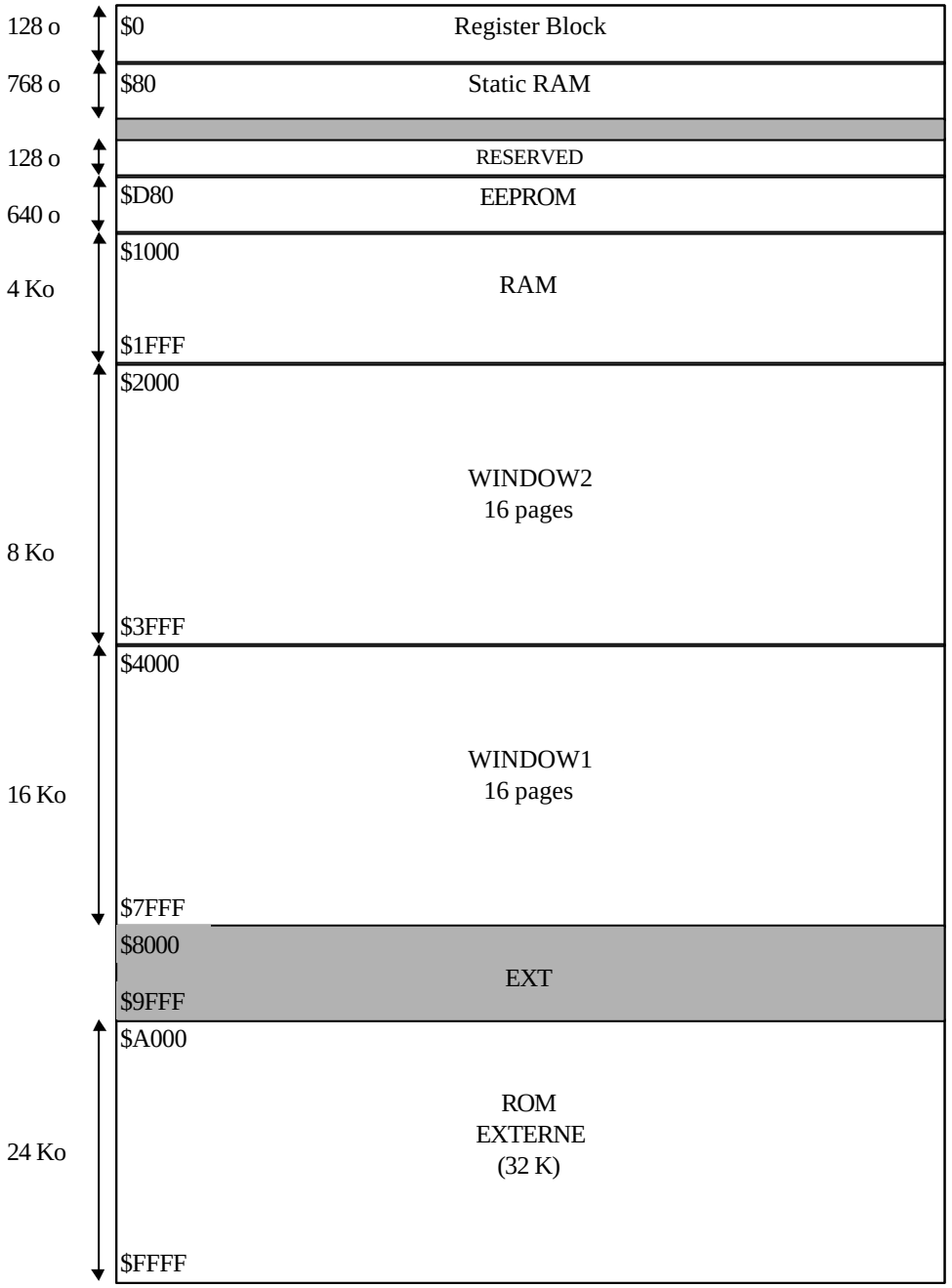


Figure 4-2 : Cartographie mémoire adressable

4.6. TRAITEMENT DES IMAGES

Il s'agit ici d'un algorithme de comparaison de deux images consécutives numérisées. Le principe est le suivant:

On dispose des 3 mémoires RAM0, RAM1 et RAM2.
Une image I_n a été numérisée. C'est l'image de référence. Les échantillons $X_{n,i}$ sont stockés dans RAM0 ou RAM1.

On numérise une image I_{n+1} et les échantillons $X_{n+1,i}$ sont stockés dans l'autre mémoire (RAM1 ou RAM0 respectivement).

On effectue alors le calcul des moyennes des échantillons:

$$M_n = M_{Ref} = \frac{1}{N}\sum_{i=0}^{N-1} X_{n,i}$$

$$M_{n+1} = M_{Test} = \frac{1}{N}\sum_{i=0}^{N-1} X_{n+1,i}$$

où N représente le nombre d'échantillons.

On effectue la comparaison des deux images I_n et I_{n+1} c'est à dire qu'on effectue la comparaison des moyennes:

$$D = \left|M_{Test} - M_{Ref}\right| = \left|M_{n+1} - M_n\right|$$

Si D est inférieur à un seuil Δ, cela signifie que la différence entre les deux images n'est pas significative. Dans ce cas, l'image I_{n+1} devient l'image de référence
($I_{ref} = I_{n+1}$) et on effectue l'acquisition d'une nouvelle image I_{n+2} qui devient la nouvelle image à tester (I_{n+2}=I_{Test}). On refait alors une nouvelle comparaison...

Si D est supérieur à Δ, alors la différence entre les deux images est significative, un signal TOR déclenche une alarme et la dernière image ,c'est à dire I_{n+1} ici, est sauvegardée dans la troisième mémoire RAM2. Il y a en fait un transfert entre les mémoires RAM0 ou RAM1 vers la mémoire RAM2. Le système se bloque et attend un nouvel ordre acquisition de la part de l'utilisateur.

L'unité de traitement

Il faut naturellement que le seuil Δ soit choisi de telle façon que le système d'alarme se déclenche pour une modification significative des images numérisées.

L'algorithme de traitement est le suivant:

On notera :
- D: différence des moyennes,
- Mref: moyenne de l'image de référence,
- Mtest: moyenne de l'image que l'on teste,
- choix: booléen qui indique si l'acquisition se fait dans la RAM0 (choix=0) ou dans la RAM1 (choix=1),
- RAM0 et RAM1: mémoires de référence et de test,
- RAM2: mémoire de sauvegarde de l'image.

```
D ← 0;
Δ ← constante;
choix ← 0;
Acquisition;
Calcul Mref;
Acquisition;
Calcul Mtest;
choix ← 1;
Répéter
        D ← valeur_absolue  (Mref-Mtest) ;
        Si D>Δ *détection* alors

                Si choix = 1 alors
                        mettre contenu RAM1 dans RAM2;
                Sinon
                        mettre contenu RAM0 dans RAM2;
                Fin de si;

                Déclencher alarme ,transfert des données de RAM2 et restitution;

        Sinon
                Mref ← Mtest;
                Si choix = 1 alors
                        sélectionner RAM1;
                        Acquisition;
                        Calcul Mtest;
```

L'unité de traitement

```
        choix ← 0;
    Sinon
        sélectionner RAM0;
        Acquisition;
        Calcul Mtest;
        choix ← 1;
    Fin de si;

Jusqu'à <alarme> ou <restitution> ou <transfert> ou <arret,pause>;
```

L'algorithme ci dessus suppose qu'on ait entre deux images consécutives une différence importante pour la détection ce qui est peu réaliste si on veut photographier l'événement d'une intrusion par exemple. On améliore alors l'algorithme en introduisant non plus un seuil Δ mais deux seuils $\Delta 1$ et $\Delta 2$ avec $\Delta 2 > \Delta 1$.

Si $D < \Delta 1$ alors on continue les acquisitions.

Lorsque $D > \Delta 1$, cela signifie qu'il y a eu une modification de l'image. On effectue alors des acquisitions en gardant comme image de référence celle qu'on avait avant le test $D > \Delta 1$. On effectue à nouveau le calcul de D jusqu'à ce qu'on obtienne soit $D < \Delta 1$, soit $D > \Delta 2$. Dans le premier cas, on recommence les acquisitions. Dans le deuxième cas, on obtient une différence d'images assez conséquente. Il y a alors mémorisation de l'image et le déclenchement d'alarme. Ainsi si, par exemple, la tête d'un délinquant passe devant la caméra, le premier algorithme mémorise seulement une partie du visage alors que l'algorithme amélioré mémorise le visage dans son ensemble.

Travail à venir

5. TRAVAIL À VENIR

Nous avons définitivement testé la partie « hardware » du projet en validant le fonctionnement du microcontrôleur, par un programme en C qui faisait uniquement une acquisition, un transfert et une restitution. Il reste donc à écrire un programme gérant le traitement et le transfert des images, ainsi que d'effectuer des tests pour trouver le seuil de détection de présence.

Les problèmes que nous avons rencontrés consistent surtout dans la réalisation de la carte. En effet, c'est un circuit mélangeant analogique et numérique. Or nous ne nous sommes pas méfiés lorsque nous avons réalisé la carte, et avons mélangé les composants analogiques et numériques. Notamment, le signal vidéo d'entrée tranversait toute une partie où il y avait beaucoup de numérique et arrivait fortement bruité à l'entrée du convertisseur analogique numérique. Nous avons dû le connecter directement à l'entrée du convertisseur pour un meilleur fonctionnement. Dans l'avenir, si une autre carte devait être réalisée et afin d'éviter tout problème d'interférence entre le numérique et l'analogique, nous conseillerions de disposer les composants comme suit :

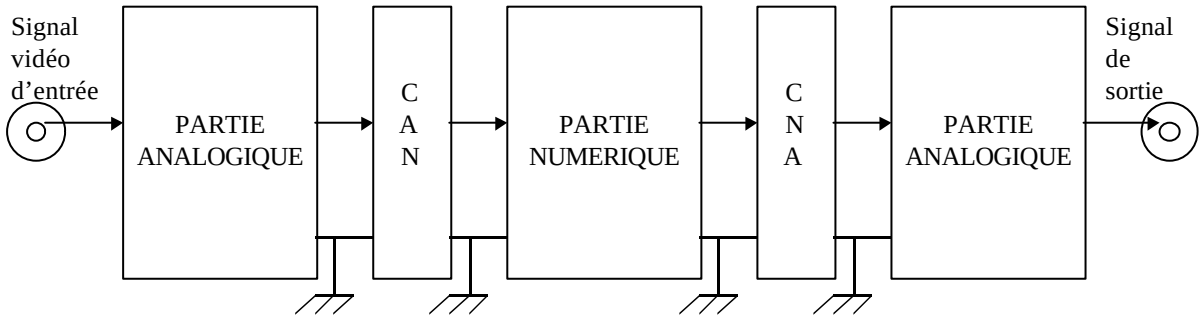


Figure 5-1 : Disposition des composants

Une autre difficulté nous est apparue lors du soudage des boîtiers plcc84 que sont les altéras et le microcontrôleur. En effet, il est absolument impossible de souder les supports sur le côté composant de la plaque. Nous avons donc soudé des barrettes de connecteurs et avons placé les supports dessus, mais il subsistait quand même des faux contacts. La solution aurait été, lors du routage, d'interdire l'accès côté composant aux pistes et de rajouter des traversées supplémentaires. Cette solution est discutable car nous ne sommes pas sûr que le routage automatique se fasse entièrement au vu des difficultés qu'il a eues avec un accès sur les deux faces.

6. CONCLUSION

La première partie de ce projet, qui consistait en la création du circuit imprimé de la carte vidéo, nous a permis, dans un premier temps, de nous familiariser avec la CAO et d'acquérir une bonne maîtrise des logiciels PCB de Mentor Graphics.

Nous nous sommes par la suite penchés sur l'étude et la validation de la carte réalisée; étude particulièrement intéressante car elle met en oeuvre de nombreux aspects de l'électronique, vus au cours de ces trois années passées à l'ENSERB : Électronique analogique, numérique, interface analogique/numérique, éléments logiques programmables, microinformatique, programmation en C, et fait également appel à certaines notions étudiées cette année dans la filière TIC, en particulier le cours de technique vidéo.

Les problèmes rencontrés, qui ont été essentiellement des problèmes d'ordre pratique (bruits, faux contacts, micro-coupure, ...), particulièrement difficiles à détecter et à résoudre, nous ont retardés dans notre projet, et ne nous ont malheureusement pas permis de le mener à son terme. La partie « hardware » est cependant bien validée.

Ce stage a donc été bénéfique sur beaucoup de points, il s'est déroulé dans de bonnes conditions, et nous ne pouvons que souhaiter l'aboutissement de ce système de surveillance.

ANNEXES

ANNEXE A : Le microcontrôleur MC68HC11K1, brochage et data sheet.

ANNEXE B : Brochage des alteras compteur et séquenceur.

ANNEXE C : Data sheet CAN AD775.

ANNEXE D : Data sheet CNA TDA8702.

ANNEXE E : Data sheet SRAM IS61C1024-20.

ANNEXE F : Data sheet ROM 27C256.

ANNEXE G : Data sheet Extracteur de synchros LM1881.

ANNEXE A : Le microcontrôleur MC68HC11K1, brochage et data sheet.

ANNEXE B : Brochage des alteras compteur et séquenceur.

ANNEXE C : Data sheet CAN AD775.

ANNEXE D : Data sheet CNA TDA8702.

ANNEXE E : Data sheet SRAM IS61C1024-20.

ANNEXE F : Data sheet ROM 27C256.

ANNEXE G : Data sheet Extracteur de synchros LM1881.